Chapter 4

Multilayer Perceptrons
Figure 4.1  Architectural graph of a multilayer perceptron with two hidden layers.
Figure 4.2  Illustration of the directions of two basic signal flows in a multilayer perceptron: forward propagation of function signals and back propagation of error signals.
Figure 4.3  Signal-flow graph highlighting the details of output neuron $j$.

The arguments $n$ indicate the iteration step, not a signal value.

Not part of the network proper; part of the training algorithm.

For each node, fan-in is a weighted sum, fan-out is identity.
**Figure 4.4** Signal-flow graph highlighting the details of output neuron $k$ connected to hidden neuron $j$. The arguments $n$ indicate the **iteration step**, not a signal value.
Figure 4.5  Signal-flow graph of a part of the adjoint system pertaining to back-propagation of error signals.

The $\delta$ are what I call “sensitivities”. The author calls them “local gradients”.

This sensitivity is determined from those at the following layer.

For each node, fan-in is a weighted sum, fan-out is identity.

The primes indicate ordinary derivatives.
Figure 4.6  Signal-flow graph illustrating the effect of momentum constant $\alpha$, which lies inside the feedback loop.

**Modification of update rule using momentum value $\alpha$.**

**Plain weight update rule**

**z-transform notation for unit delay.**
Figure 4.7  Signal-flow graphical summary of back-propagation learning. Top part of the graph: forward pass. Bottom part of the graph: backward pass.

Note the use of derivatives at forward activations for computing backpropagated error.

The $\delta$ are “sensitivities”.

output values  \( o \)
desired values  \( d \)
Figure 4.8  (a) Architectural graph of network for solving the XOR problem.  (b) Signal-flow graph of the network.
Figure 4.9  (a) Decision boundary constructed by hidden neuron 1 of the network in Fig. 4.8. (b) Decision boundary constructed by hidden neuron 2 of the network. (c) Decision boundaries constructed by the complete network.

Function of hidden neuron 1

Function of hidden neuron 2

Function of entire network
Figure 4.10  Graph of the hyperbolic tangent function \( \phi(v) = \tanh(bv) \) for \( \alpha = 1.7159 \) and \( b = \frac{2}{3} \). The recommended target values are +1 and −1.

See p. 145. \( \alpha \) and \( \beta \) are from LeCun, 1993.

Keeping the targets away from saturation range helps prevent stagnation in training.
Figure 4.11  Illustrating the operation of mean removal, decorrelation, and covariance equalization for a two-dimensional input space.


Applying these transformations to the training set means they must also be applied to the test set.

- Subtract means
- Scaling by variances
- Mean removal
- Decorrelation
- Covariance equalization

Can be done with PCA (later in course)
Figure 4.12  Results of the computer experiment on the back-propagation algorithm applied to the MLP with distance $d = -4$. MSE stands for mean-square error.

“Double moon” data set (parameter $d = -4$ vertical separation).

Final error 0.

20 neurons in hidden layer

hyperbolic tangent

Learning rate annealed linearly $10^{-1}$ to $10^{-5}$. 
Figure 4.13  Results of the computer experiment on the back-propagation algorithm applied to the MLP with distance $d = -5$.

“Double moon” data set (parameter $d = -5$ vertical separation).

Final error 0.0015
Figure 4.16  (a) Properly fitted nonlinear mapping with good generalization. (b) Overfitted nonlinear mapping with poor generalization.

Good generalization

Overfitting to training data
Figure 4.17  Illustration of the early-stopping rule based on cross-validation.

Using validation data to avoid overfitting
Figure 4.18  Illustration of the multifold method of cross-validation. For a given trial, the subset of data shaded in red is used to validate the model trained on the remaining data.

Cross-Validation:

Shaded data is validation data.

Rotation is through multiple complete trainings.
Matrix Formulation

• Each neuron has a **weight vector**.
• A **layer** is a vector of neurons, so a layer has a **weight matrix**, the rows of which are the weight vectors of each neuron.